Proyecto Final DevOps: PetClinic

Patricia Fernández Caballero

Bootcamp DevOps II Código Facilito

Contenido

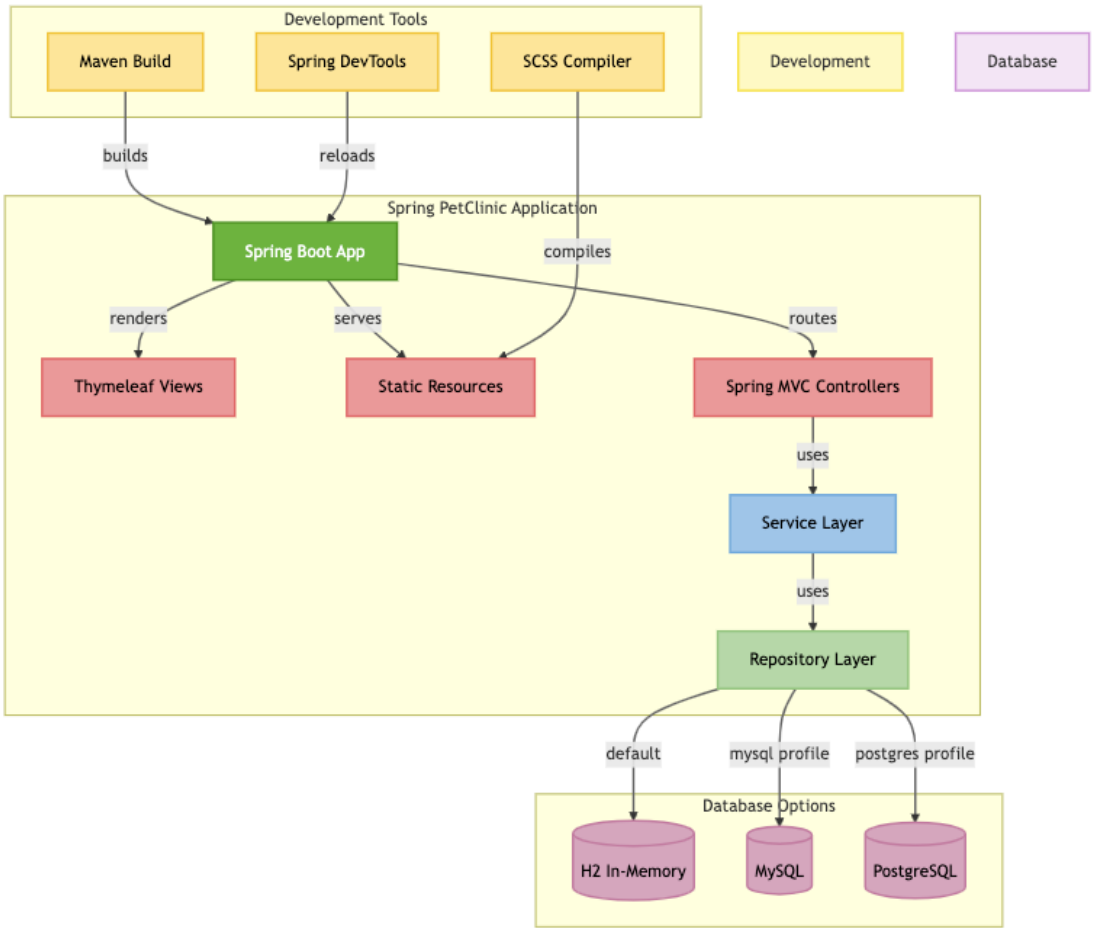
Introducción y Guía General

¿Qué es PetClinic?

PetClinic es una aplicación web construida con Spring Boot que simula la gestión de una clínica veterinaria. Permite registrar dueños, mascotas y sus visitas mediante una interfaz web intuitiva. Utiliza una arquitectura basada en el patrón MVC, persistencia con Spring Data JPA y una base de datos relacional MySQL.

Esta aplicación es usada ampliamente con fines educativos para aprender el ecosistema Spring y se ha convertido en un proyecto de referencia para prácticas DevOps modernas.

Arquitectura General

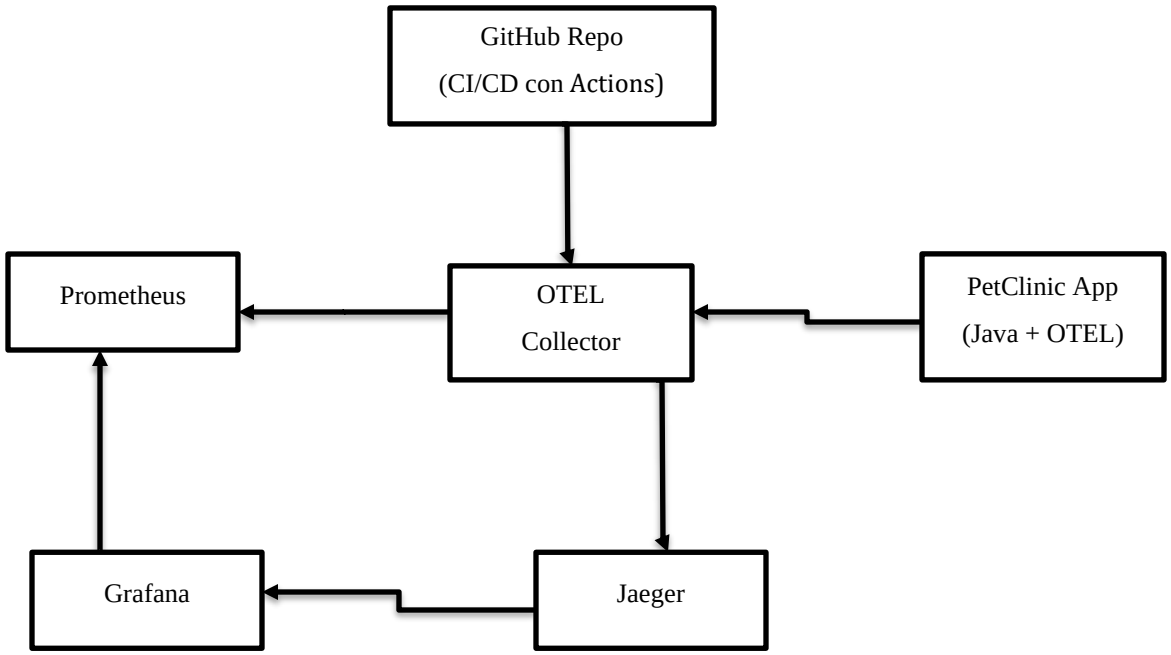


Datos técnicos

Requisitos Previos

- Docker y Docker Compose instalados
- Git
- Helm
- Minikube
- Cuenta de GitHub
- Cuenta de Docker Hub (para CI/CD)

Arquitectura



Tecnologías Utilizadas

- Java 21 (Spring Boot)
- Maven
- Docker
- Docker Compose
- Prometheus
- Grafana
- Jaeger
- OpenTelemetry
- GitHub Actions
- Helm
- Minikube (Kubernetes local)
- Trivy

Guía de Despliegue

Paso a describir los pasos necesarios para desplegar la aplicación PetClinic en dos entornos: Docker Compose y Kubernetes con Helm (Minikube). Así como el uso de GitHub Actions y de Trivy para seguridad.

Despliegue con Docker Compose

1. Clonar el repositorio

```
git clone https://github.com/PattFC/petclinic.git

cd petclinic
```

2. Iniciar stack de monitorización

```
docker compose -f docker-compose.monitor.yml up -d
```

3. Iniciar la aplicación

```
docker compose -f docker-compose.yml up -d --build
```

4. Verificar servicios

- PetClinic: http://localhost:8080
- Prometheus: http://localhost:9090
- Grafana: http://localhost:3000
- Jaeger: http://localhost:16686

Despliegue con Helm + Minikube

1. Iniciar Minikube

```
minikube start --driver=docker
```

2. Aplicar MySQL como servicio interno

```
kubectl apply -f helm/k8s/mysql-deployment.yaml
```

3. Desplegar la aplicación con Helm

```
helm install petclinic ./helm
```

4. Acceder a la aplicación

```
kubectl port-forward service/petclinic-svc 8081:8080
```

Abrir en navegador: http://localhost:8081

CI/CD con GitHub Actions

El repositorio incluye un pipeline GitHub Actions que:

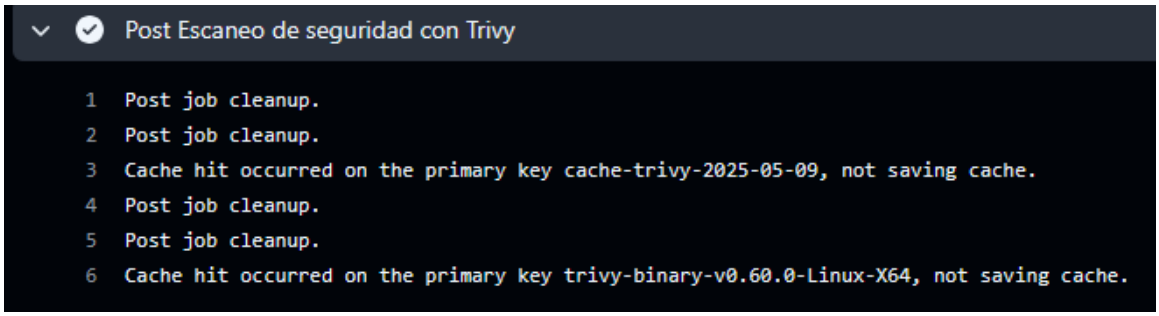
- Compila y testea la aplicación con Maven
- Construye una imagen Docker
- Inicia sesión en Docker Hub
- Etiqueta y sube la imagen como `latest`

La imagen final puede encontrarse en: https://hub.docker.com/r/pattfc/petclinic/tags

Seguridad: Análisis de vulnerabilidades con Trivy

Este proyecto incluye un análisis automático de seguridad en el pipeline CI/CD. Se utiliza Trivy para escanear la imagen Docker en busca de vulnerabilidades críticas o altas antes de subirla a Docker Hub.

Resultado visible en la pestaña Actions (GitHub)



Paso relevante en `.github/workflows/ci.yml`:

```
- name: Escaneo de seguridad con Trivy
  uses: aquasecurity/trivy-action@master
  with:
      scan-type: image
      image-ref: petclinic-app
      format: table
      severity: CRITICAL,HIGH
```

Runbook de Incidentes

En este apartado recojo respuestas rápidas para resolver incidencias comunes en el entorno DevOps del proyecto PetClinic.

Aplicación no conecta a MySQL

Diagnóstico

- Revisar los logs del pod:

```
kubectl logs deployment/mysql

kubectl logs deployment/petclinic-app
```

Solución

- Revisar que el pod `mysql` está corriendo.

- Verificar credenciales en el `deployment.yaml`.

El pod `petclinic-app` entra en CrashLoopBackOff

Diagnóstico

```
kubectl get pods

kubectl logs <nombre-del-pod>
```

Solución

- Esperar a que MySQL esté listo antes de instalar la app.

- Reinstalar Helm chart si es necesario:

```
helm uninstall petclinic

helm install petclinic ./helm
```

Port-forward falla (puerto ocupado)

Diagnóstico

```
lsof -i :8080
```

Solución

- Usar otro puerto:

```
kubectl port-forward service/petclinic-svc 8081:8080
```

CI/CD falla al subir imagen

Diagnóstico

- Ver error en GitHub Actions → pestaña “Actions”

Solución

- Revisar que `DOCKER_USERNAME` y `DOCKER_PASSWORD` estén correctamente configurados en GitHub Secrets.

- En este caso había conflicto de mayúsculas en el username.

- Volver a hacer `git push`.

Observabilidad y Monitorización

En este apartado describo brevemente cómo se ha implementado la observabilidad de la aplicación PetClinic en el entorno del proyecto DevOps.

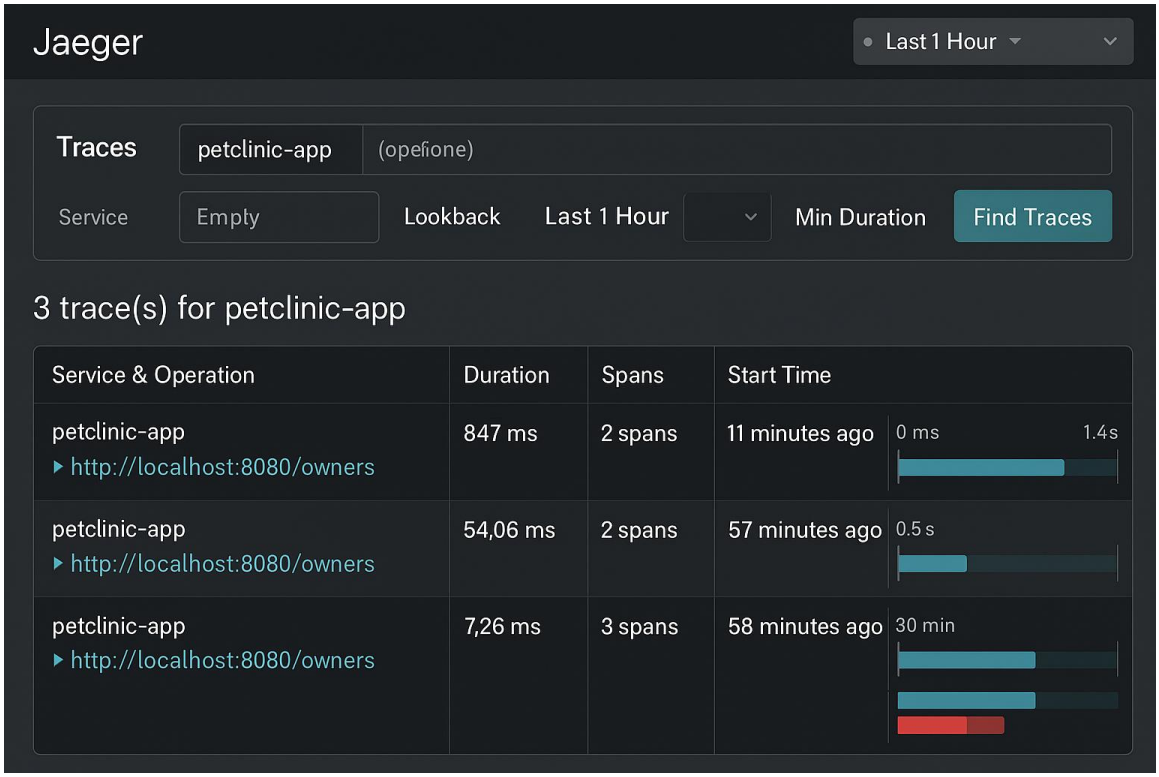
Herramientas involucradas:

- **Jaeger**: trazas distribuidas
- **Prometheus**: recolector de métricas
- **Grafana**: visualización de métricas
- **OpenTelemetry**: instrumentación de la app Java
- **Spring Actuator**: exposición de métricas y endpoints internos

Jaeger

Jaeger permite visualizar trazas distribuidas para entender cómo se comportan las llamadas dentro de la aplicación.

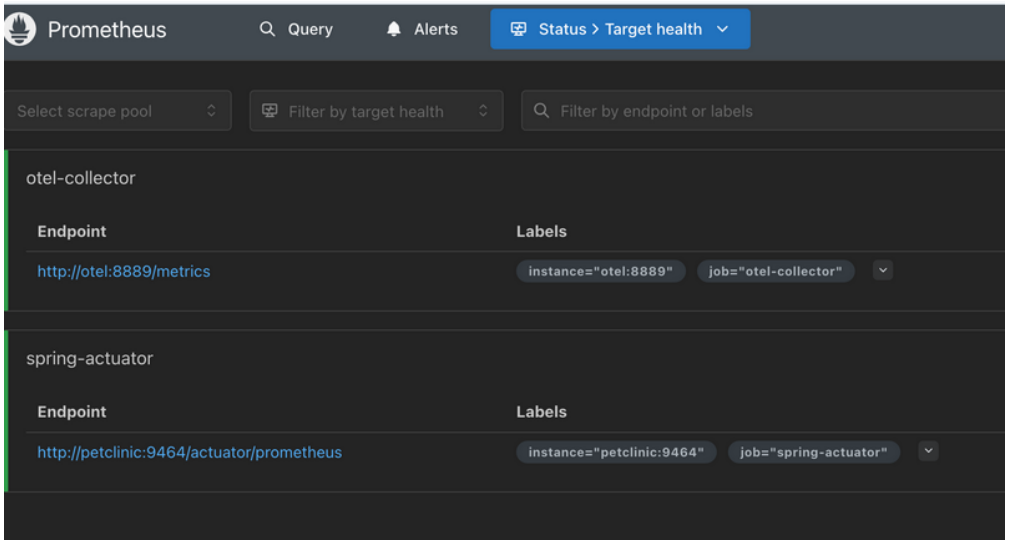
- Recibe trazas desde el agente OTEL
- Visualiza el flujo entre controladores y servicios de Spring Boot
- Muy útil para detectar latencias o errores distribuidos



Prometheus

Prometheus se encarga de recolectar métricas de la aplicación y exponerlas para visualización.

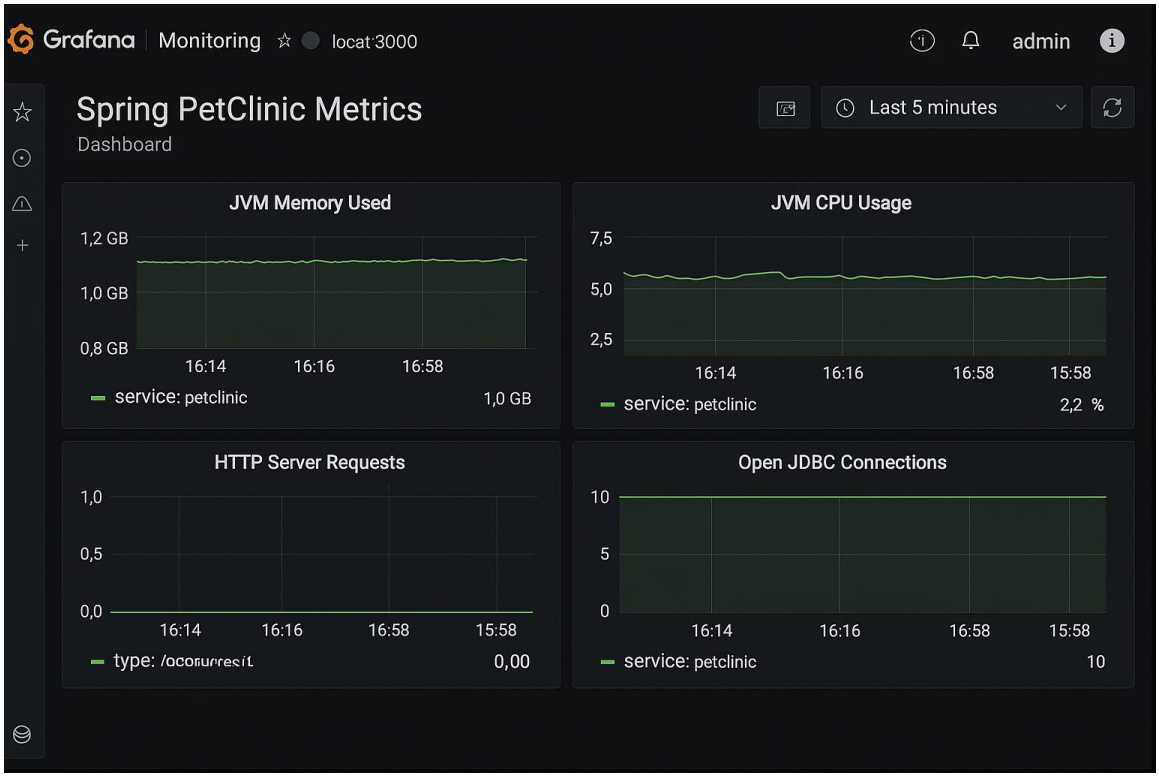
- Las métricas se obtienen vía OpenTelemetry (OTLP) y Spring Actuator.
- Puerto expuesto por el collector: `8889`
- Las métricas pueden incluir:
    - Recuento de peticiones HTTP
    - Duración promedio de peticiones
    - Estado de endpoints



Grafana

Grafana se ha configurado para mostrar paneles visuales con métricas clave del sistema.

- Se utiliza Prometheus como fuente de datos
- Paneles incluyen:
- Recuento de peticiones HTTP
- Duración promedio
- Por endpoint y método



Observaciones

Las herramientas de observabilidad se pueden iniciar con:

```
docker compose -f docker-compose.monitor.yml up -d
```

Todos los servicios están asociados a una red Docker llamada `petclinic-net`.